ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Technique for Keyword Search using Minimum Distance

Chavan Aparna[1], S.B. Bangar[2]

M. Tech Scholar, Dept. of Computer Science & Engineering, Maharashtra Institute of Technology (MIT), Aurangabad, Maharashtra, India[1]

Assistant Professor, Dept. of Computer Science & Engineering, Maharashtra Institute of Technology (MIT), Aurangabad, Maharashtra, India[2]

ABSTRACT: In documents the Keyword search is the most useful information discovery method. Now a day's databases having a large amount of data. Efficient processing of top-k queries is a important requirement in the plain text which coexists with unstructured data, structured data. This paper describes the keyword search with single keyword search and two keywords search with the finding of minimum distance using Levenshtein distance. This paper includes relational database, top-k querying using the ranking method and structural relationship using indexing for keyword search. Recently, for improving the keyword search and index size used for structural relationship it is done by joining the multiple relevant Tuple units. In literature survey various different existing techniques are studied. This survey also describes the Ranking method. Ranking queries are effective in many rising applications for finding top-k answers. The research methodology used to resolve is top-k query processing. In proposed system Lucene search library is used. Lucene is simple yet powerful java based search library. There are mainly two kinds of XML programming interfaces, SAX (Simple API for XML) and DOM (Document Object Model). The main focus is on SAX parser. In proposed work computing the Levenshtein distance is established on the consideration that if we reserve a matrix to hold the Levenshtein distance between all prefixes of the main string and all prefixes of the second, then in a dynamic programming we can figure the qualities in the grid and in this manner processing the separation between the two full strings as the last esteem registered. We have implemented our method using the database systems and java programming language and the experimental results show that our approach achieves high result quality. It is used in order to improve the response time efficiency, accuracy and simplicity of the design process.

KEYWORDS: Keyword Search; Relational Database; XML parsing; Levenshtein distance; Lucene

I. INTRODUCTION

Data mining is the mechanism that attempts to discover patterns in large volume of data sets. The methods used by the Data mining are at the junction of machine learning, statistics, an artificial intelligence and database systems. From a data set extracting the information and transform it into an understandable structure for further use is the main aim of the data mining process. Emerging applications that rely upon ranking queries warrant efficient support of ranking in database management systems. Recently, the integration of database and information retrieval (IR) technologies has been an active research topic. [1]

Querying structured data over Relational databases is a repository for a significant amount of data (e.g. enterprise data) and RDBMS managing an abstract view of underlying data. And the Structured Query Language (SQL) is precise and complete which is difficult for the casual users. The relational databases are regularly searched using structured query languages. The internet user wants to get the correct answers quickly and efficiently. The clients are not pleasing to peruse through the complete answer set. Watchword hunt is a broadly acknowledged component for questioning in printed archive frameworks and World Wide Web. Introducing keyword search capability into relational databases XML databases, graph databases and heterogeneous data sources the database research community has recently recognized the benefits of keyword search.[1][2][3][4]

Keyword search by structured and semi-structured data requires collecting together the data from different locations

DOI: 10.15680/IJIRCCE.2016. 0404239

which are interconnected and collectively relevant to query. Keyword searching over relational databases obtains the answers of tuples in the databases that are through Primary/Foreign key and contain query keywords. Ongoing studies can be generally divided into three kinds of terminologies: candidate-network-based methods [5], Steiner-tree-based algorithms [7] and tuple unit-based approaches [2][3]. The XML has a tree structure in which result is a sub tree established at the most minimal regular predecessor of an arrangement of hubs and all things considered matches query keywords [4][14].

Tuple units are effective to answer keyword queries as they catch structures and can represent a significant, relevant and integral information unit [1][3].

1. The Tuples associated through essential/outside keys connections that can be distinguished and listed, and in this way we can effectively answer keywords inquiries by utilizing such ordered auxiliary data.

2. The quantity of tuple units won't be huge, which is not bigger than that of the aggregate tuples in the basic database. In the event that every estimation of the essential key is eluded by the remote key, the quantity of tuple units is the same as the quantity of tuples in the tables with outside keys. By and by, the quantity of tuple units is much littler than the aggregate number of tuples in the basic database as tentatively demonstrated [3].

3. We can utilize database abilities to produce and emerge the tuple units by making a perspective on top of the fundamental social tables. We require no extra records to keep up tuple units [1].

Java gives a broad arrangement of set of tools for manipulating character-based information, including byte-to-character conversion or transformation, input and output streams including sockets and compression, memory-mapping files and records, and high-level methods for manipulating sequences of characters like buffers and regular expressions. It likewise cover the utilization of the Apache Lucene search library [8][9].

II. Related work

2.1 What is Keyword Search?

The term Keyword Search is that a user presents or submits a query using a finite set of keyword to discover the information that fulfills his/her information needs [1]. The advantages of the keyword search on databases are: It is easy to use and permit the interesting or unexpected discoveries to access the data in web and scientific applications. That is relevant data are spread but are rightly relevant to query should be automatically gather in the result. Keyword search over structured and semi-structured data required to assemble together data from different places which are interconnected and collectively related to query. [1][3]

2.2 Information Retrieval

The user’s objective is focused on finding documents, record sub-elements, summaries, or surrogates that are related to a query this comes within the context of information retrieval (IR). This may be an iterative method, with human response or feedback, but it is normally limited to a single session. Conventional IR tasks include finding documents with terms that match terms presented by the keyword/document searcher, or finding relevant data or resources related to a query. Typically, within each IR task, the keyword/document searcher indicates the queries systematically, inspects or examines results, and chooses distinct documents to view. As a result of examining search results and viewing documents, searchers gathers information to help and to satisfy their actual information-seeking issue and eventually the higher-level information need. [8]

2.3 Top-k Query

The meaning of concept query is a finite set of keywords. For clear meaning of top-k query define as- Given a database D of p objects, each of which is to define the character by n attributes, a scoring function f, in concurrence with to which we rank the object of the database D. Then a top-k query Q retrieve the k objects with the highest rank in f efficiently. Rather than all of the answers, a top-k query returns the subset of most relevant answers. [1][3]

2.4 Tuple Units

Given a database with m connected tables, the tuples (records) that can be coordinated together through the primary/foreign keys must be exceptionally relevant with each other. Tuple units are proficient to answer keyword queries and can represent meaningful and integral information units. The database is modelled as a directed graph.

Tuples → nodes.
Foreign-key, Primary-key link → edge.

Result is a rooted directed tree including at least one node having each query keyword. A tuple unit is an arrangement of set of highly relevant tuples which contain query keywords. We can use indexed tuple units to efficiently answer a keyword query.[3][4]

2.5 Steiner-tree-based search

A relational database can be demonstrated as a database graph G = (V, E) such that there is a coordinated mapping between a tuple in the database and a node in V. G expected to be as a directed graph with two a edge: a forward edge (u, v) ∈E if and only if there is a foreign key from u to v, and a back edge (v, u) if and only if (u, v) is a forward edge in E. An edge (u, v) proposes a close relationship between tuples u and v. [1] [3][12][13]

Most of existing strategies of keyword search over relational databases discover the Steiner trees made out of related tuples during process of answers. The Steiner trees are recognized by discovering the rich structural relationships amongst tuples, and avoid the fact that such structural relationships can be pre-processed and indexed. Tuple units that are made out of most relvant tuples are proposed to address this issue. Tuple units can be computed and indexed. [5][12][13]

2.6 XML Search

An adaptable approach to build information formats and electronically share structured organized information by means of people in general Internet, and also through corporate systems is a XML standard. A9hierarchical format for data exchange and representation is the eXtensible Markup Language (XML). An XML document comprises of nested XML elements beginning with the root element. Every element can have attributes and values, in extra to nested sub-elements. In the form of XML documents it is becoming more popular to publish data on the Web. Present search engines, which are a essential tool for discovering HTML documents, have two main downsides when it meet expectations to searching for XML documents. In the first place, it is unrealistic to pose queries that definitely cite to meta-data (i.e., XML tags). Hence, it is difficult, to specify systematically a search query that combine semantic knowledge in a clear and accurate way. The web search tools return references (i.e., links) to documents and not to particular fragments that is the second drawback. This is risky, since substantial XML documents (e.g., the XML DBLP) may contain Permission to duplicate without charge all or portion of this material is granted provided that the duplicates are not made or dispersed for direct commercial advantage [2] [11] For instance, a creator is identified with titles of papers she composed, yet not to titles of different papers. Really, if a web searcher essentially coordinates the hunt terms against the archives, it might return records that don't answer the client's question. This happens when unmistakable hunt terms are coordinated to disconnected parts of a XML archive, as showed in the following case.[25]

The essential approaches to parse XML are DOM and SAX parsers:

- A DOM parser is a Document Object Model. For XML information it delivers an in-memory tree representation. This is pleasant for little or medium measured XML records, however it is not commonsense for a more noteworthy than 400M report.

- A SAX parser is a Simple API for XML. It underpins a lower level get back to interface. The strategies 'startElement', "endElement" and "charcters" are called if an open label, end tag, or any characters between the labels are perceived

III. PROPOSED ALGORITHM

For example we model the graph in Fig.1 as a BAM where the characters denote the keywords contained in the corresponding unit. Take this graph as a running example throughout the report, in this matrix the cell for tuple unit u_2 and tuple unit u_4 is 1 as there is an edge between the two tuple units given in graph. Different from BAM the value at the i^{th} row and j^{th} column of MDM is the minimal distance of two tuple units u_i and u_j. In order to preserve the paths between two nodes with minimal distance, we introduce another matrix, minimal path matrix. The entry of $<u_i, u_j>$ in MPM preserves the path accompanying minimal distance between u_i and u_j. For example we construct MDM and MPM of the graph. In MDM, the value entry $< u_4, u_1>$ is 2. This means that the minimal distance from u_4 to u_1 is 2. In MPM, the value entry $< u_4, u_1>$ is u_2. That is the minimal path from u_4 to u_1 is u_4-u_2-u_1. [1][3]

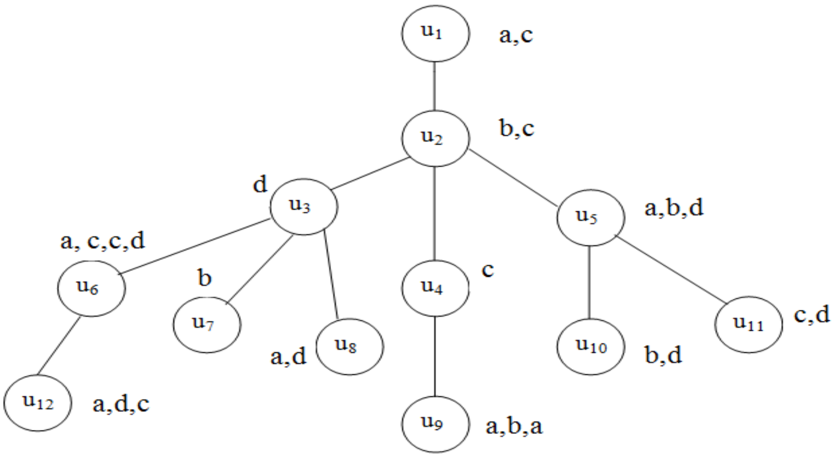

Fig.1. Graph Example

3.1 Term Frequency

This Term Frequency modeled every tuple unit as a document and takes the terms in the tuple units as keywords

- u- Set of tuple units
- p- distinct tuple units
- q- Keywords in u.
- k_i-Keyword in tuple unit u.

tf(k_i, u) as the term frequency of k_i in u, which is the number of occurrence of k_i in u.the term frequency is calculated for each and and every tuple units which contained the four keywords that is a,b,c,d.[26][27]

3.2 Inverse Document Frequency

We denote idf(k_i) as the inverse document frequency of [2] ;

$idf(k_i) = (p+1)/ (O_{ki}+ 1)$

Where

- p- Distinct tuple units
- O_{ki} - Number of such tuple units which directly contain k_i [26][27]

3.3 Normalized Term Length

We denote (u) as the normalized term length of u and p is distinct tuple units calculated out, where**,**

$ntl(u) = |u|/ (1/p) * \sum |u'|$

Where, |u|- Denotes the number of terms in u.

DOI: 10.15680/IJIRCCE.2016. 0404239

3.4 Levenshtein Distance

The Levenshtein separation between two strings is the base number of operations expected to change one string into the other, where an operation might be insertion, cancellation or substitution of one character. - might be insertion, cancellation or substitution of one character.

Levenshtein separation (LD) is a measure of the likeness between two strings, which we will allude to as the source string (c) and the objective string (o). The separation is the quantity of cancellations, insertions, or substitutions required to change c into o [7][10]. For instance,

- If c is "test" and o is "test", then LD(c,o) = 0, in light of the fact that no changes are required. The strings are now indistinguishable.
- If c is "test" and o is "tent", then LD(c,o) = 1, since one substitution is adequate to change c into o.

The more noteworthy the Levenshtein separation, the more diverse the strings are. The Levenshtein Distance, likewise regularly alluded to as the alter separation is defined as the insignificant number of progress operations, that are expected to change a String c into a String o.

The reasonable change operations are

- insertion of a character,
- deletion of a character and
- replacement of a character.

Case in point, we may change the String "period" into "pearls" utilizing 1 character insertion, 2 character substitutions and 1 character erasure, consequently bringing about a Levenshtein separation of 4 [35]. The idea of the Levenshtein separation is genuinely straightforward and has as of now been clarified. In any case, there are a couple fascinating qualities of the Levenshtein separation that ought to be called attention to.

1. Lower Bound

We should see that there is a lower headed for the Levenshtein separation.

2. Upper Bound

Likewise, there is additionally an upper headed for the Levenshtein separation between two strings c and o [7][23].

3.5 Lucene

Lucene is a full-message seek library in Java which makes it simple to add look usefulness to an application or site. It does as such by adding substance to a full-message list. It then permits you to perform inquiries on this record, returning results positioned by either the importance to the question or sorted by a subjective field, for example, a report's last changed date. The substance you add to Lucene can be from different sources, similar to a SQL/NoSQL database, a filesystem, or even from sites.

Seeking and Indexing

Lucene can accomplish quick pursuit reactions in light of the fact that, rather than looking the content straightforwardly, it seeks a record. This would be what might as well be called recovering pages in a book identified with a catchphrase via looking the record at the back of a book, rather than seeking the words in every page of the book. This sort of file is called a rearranged file, since it transforms a page-driven information structure to a Keyword driven information structure [17][23] .

IV. Pseudo code

```
Step 1: Initialization
a) Initialize n to be the length of s, set m to be the length of t.
b) Construct a grid containing 0..m lines and 0..n sections.
c) Initialize the principal line to 0..n,
d) Initialize the primary segment to 0..m.
```

Step2: Processing
a) Examine s (i from 1 to n).
b) Examine t (j from 1 to m).
c) If s[i] squares with t[j], the expense is 0. d) If s[i] doesn't approach t[j], the expense is 1.
e) Set cell d[i,j] of the lattice equivalent to the base of:
i) The cell quickly above in addition to 1: d[i-1,j] + 1.
ii) The cell promptly to one side in addition to 1: d[i,j-1] + 1.
iii) The cell askew above and to one side in addition to the expense: d[i-1,j-1] + cost.

Step 3: Result
Step 2 is rehashed till the d[n,m] quality is found.
Step 4: End

V. Simulation Results

The "trueness" or the closeness of the diagnostic result to the "genuine" quality. It is constituted by a mix of irregular and methodical blunders (accuracy and predisposition) and can't be evaluated straightforwardly. Here in this section in fig 2. The proposed keyword search is implemented in Java using the Levenshtein distance approach the top-10 and top-50 and top-100 results are taken and other existing systems result values are taken from the paper existing paper that is studied SAINT-SKSA and SAINT-KPSA and it shows the better keyword search accuracy. Fig 3. Shows better response time than existing systems. It has the top-10, top-50 and top-100 keyword search results are found in time that is in milliseconds. JavaFX is utilized to ascertain reaction time. JavaFX is an arrangement of illustrations and a media bundle that empowers engineers to plan, make, test, troubleshoot, and send rich customer applications that work reliably crosswise over differing stages.

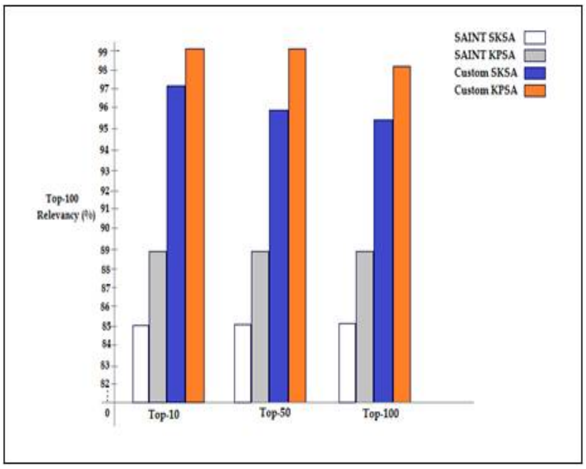

Fig.2. Search Accuracy

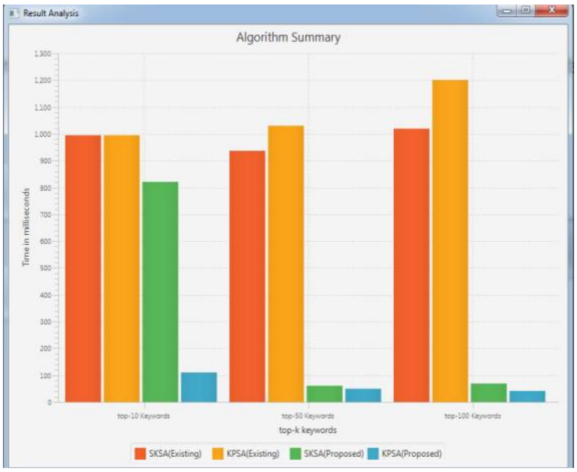

Fig. 3. Response Time

VI. Conclusion and Future Work

It describes the multiple terms for efficient searching of keywords by integrating several related tables with tuple units, by using structure aware indexes we also studied the related tuples are connected through the primary-foreign-key relationships by retrieving the Steiner trees or by calculating candidate networks on the fly. Indexer gets the information from archive and construct a reversed file to bolster keywords based looking (like a traditional search engine search keywords in various documents). The experimental results show that, this given approach achieves high search accuracy, Search efficiency and response time of the system. To support keyword based searching, query processing method and Lucene seek utilizing levenshtein distance has been proposed in this stage it splits the user query into keywords then searches these keywords in inverted index for attribute name.

REFERENCES

1. Jianhua Feng, Member, Guoliang Li, and Jianyong Wang, “Finding Top-k Answers in Keyword Search over Relational Databases Using Tuple Units” The authors are with the Department of Computer Science and Technology, Tsinghua University. Vol.12, pp.12, Dec.2011
2. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, “Xrank: Ranked Keyword Search over XML Documents,” Proc. ACM SIGMOD Int’l Conf. Management of Data, pp. 16-27, 2003.
3. G. Li, J. Feng, and L. Zhou, “Retune: Retrieving and Materializing Tuple Units for Effective Keyword Search over Relational Databases,” Proc. Int’l Conf. Conceptual Modeling (ER), pp. 469-483, 2008.
4. V. Hristidis and Y. Papakonstantinou, “Discover: Keyword Search in Relational Databases,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 670-681, 2002.
5. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, “Keyword Searching and Browsing in Databases Using Banks,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 431-440, 2002.
6. B. Ding et al., “Finding Top-k Min-Cost Connected Trees in Databases,” Proc. IEEE Int’l Conf. Data Eng. (ICDE), 2007.
7. Rishin Haldar and Debajyoti Mukhopadhyay, “Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach” Web Intelligence & Distributed Computing Research Lab Green Tower, C-9/1, Golf Green, Calcutta 700095, India.
8. G. Salton and M. J. McGill. "Introduction to modern information retrieval". 1983.
9. Effective semantic-based keyword search over relational databases for knowledge discovery Sina Fakhraee Wayne State University, 2012.
10. Stavros Konstantinidis Department of Mathematics and Computing Science Saint Mary’s University Halifax, Nova Scotia B3H “Computing the edit distance of a regular language”. Volume 205, Issue 9, September 2007, Pages 1307–1316
11. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, “Xsearch: A Semantic Search Engine for XML,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 45-56, 2003.
12. S. Agrawal, S. Chaudhuri, and G. Das, “DBXplorer: A System for Keyword-Based Search over Relational Databases,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 5-16, 2002.
13. L. Qin, J.X. Yu, and L. Chang, “Keyword Search in Databases: The Power of RDBMS,” Proc. SIGMOD Int’l Conf. Management of Data, pp. 681-694, 2009.
14. H. He, H. Wang, J. Yang, and P. Yu, “Blinks: Ranked Keyword Searches on Graphs,” Proc. ACM SIGMOD Int’l Conf. Management of Data, 2007
15. V. Hristidis, L. Gravano, and Y. Papakonstantinou, “Efficient Ir-Style Keyword Search over Relational Databases,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 850-861, 2003
16. A. Balmin, V. Hristidis, and Y. Papakonstantinou, “Objectrank:Authority-Based Keyword Search in Databases,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 564-575, 2004.
17. Q. Su and J. Widom, “Indexing Relational Database Content Offline for Efficient Keyword-Based Search,” Proc. Int’l Database Eng. and Application Symp. (IDEAS), 2005.
18. B. Kimelfeld and Y. Sagiv, “Finding Approximating Top-k Answers in Keyword Proximity Search,” Proc. ACM SIGMODSIGACT-SIGART Symp. Principles of Database Systems (PODS), 2006.
19. G. Li, B.C. Ooi, J. Feng, J. Wang, and L. Zhou, “Ease: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data,” Proc. ACM SIGMOD Int’l Conf. Management of Data, pp. 903-914, 2008.
20. F. Liu, C. Yu, W. Meng, and A. Chowdhury, “Effective Keyword Search in Relational Databases,” Proc. ACM SIGMOD Int’l Conf. Management of Data, pp. 563-574, 2006.
21. Y. Luo, X. Lin, W. Wang, and X. Zhou, “Spark: Top-k Keyword Query in Relational Databases,” Proc. ACM SIGMOD Int’l Conf. Management of Data, 2007.
22. Y. Xu and Y. Papakonstantinou, “Efficient Keyword Search for Smallest LCAs in XML Databases,” Proc. ACM SIGMOD Int’l Conf. Management of Data, pp. 527-538, 2005.
23. G. Li, J. Feng, and J. Wang, “Structure-Aware Indexing for Keyword Search in Databases,” Proc. ACM Conf. Information and Knowledge Management (CIKM), pp. 1453-1456, 2009.
24. https://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html.
25. Sushma. J. Basanagoudar, Dr. B. G. Prasad IJCAT International Journal of Computing and Technology “Interactive Fuzzy based Search over XML Data for Optimized Performance”, Volume 1, Issue 6, July 2014 ISSN : 2348 – 6090
26. G. Salton and C. Buckley. "Term-weighting approaches in automatic text retrieval". Information Processing & Management, 24 (5). 1988.
27. H. Wu and R. Luk and K. Wong and K. Kwok. "Interpreting TF-IDF term weights as making relevance decisions". ACM Transactions on Information Systems, 26 (3). 2008.
28. Chavan Aparna “Review on keyword search over relational databases “Volume 4, Issue 11, November 2014”.
29. http://dblp2.uni-trier.de/db.
30. Ilyas IF, Aref WG, Elmagarmid AK (2003) Supporting top-K join queries in relational databases. In: Proceedings of the 29th international conference on very large data bases, pp 754–765, September 9–12,2003, Berlin, Germany.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

BIOGRAPHY

S. B. Bangar is a Assistant Professor in the Department of Computer Science & Engineering, Maharashtra Institute of Technology (MIT), Aurangabad, Maharashtra. Her research interests are Data Mining, Stegnography.

A. R. Chavan is a Research Student in the Department of Computer Science & Engineering, Maharashtra Institute of Technology (MIT), Aurangabad, Maharashtra. She received Master of Technology degree in 2016 from BAMU, Aurangabad, MS, India. Her research interests are Data Mining, Relational Database.